1999

Mon. Not. R. Astron. Soc. **000**, 1–9 (1999)

A TreePM code for Cosmological N-Body simulations

J.S.Bagla⋆
Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, MS-51, Cambridge MA 02138, U.S.A.

Oct.25, 1999

ABSTRACT
We describe a hybrid technique for carrying out large N-Body simulations to study formation and evolution of the large scale structure in the Universe. This hybrid code, called the treePM code, is a combination of Barnes and Hut (1986) tree code and Particle-Mesh code. Such a code combines the speed of PM simulations and the automatic inclusion of periodic boundary conditions with the high resolution of tree codes. We describe the splitting of force between the PM and the tree part, and estimate errors in force for various possible combinations. We use the error analysis to suggest an optimum configuration of the code. We present some example applications to estimate the efficacy of this code for realistic applications. We also discuss the possibility of parallelising this code.

Key words: gravitation, methods: numerical, cosmology: large scale structure of the universe

1 INTRODUCTION

Observations suggest that the present universe is populated by very large structures like galaxies, clusters of galaxies etc. Current models for formation of these structures are based on the assumption that gravitational amplification of density perturbations resulted in the formation of large scale structures. In absence of analytical methods for computing quantities of interest, numerical simulations are the only tool available for study of clustering in the non-linear regime. The last two decades have seen a rapid development of techniques and computing power for cosmological simulations and the results of these simulations have provided valuable insight into the study of structure formation.

The simplest N-Body method that has been used for studying clustering of large scale structure is the Particle Mesh method. It has two elegant features in that it provides periodic boundary conditions by default, and the force is softened naturally so as to ensure collisionless evolution of the particle distribution. However, softening of force done at grid scale implies that the force resolution is very poor. In particular, this limits the dynamic range over which we can trust the results of the code between a few grid cells and about a quarter of the simulation box (Bouchet and Kandrup, 1985; Bagla and Padmanabhan, 1997). Many efforts have been made to get around this problem, mainly in the form of P^3M (Particle-Particle Particle Mesh) codes (Efstathiou et al, 1985; Couchman, 1991). In these codes, the force computed by the particle mesh part of the code is supplemented by adding the short range contribution of nearby particles. The main problem with this approach is that this code gets bogged down in the particle-particle part in highly clustered situations.

A completely different approach to the problem of computing force are codes based on the tree method. In this approach we consider groups of particles at a large distance to be a single entity and compute the force due to the group rather than sum over individual particles. There are different ways of defining a group, but by far the most popular method is that due to Barnes and Hut (1986). Applications of this method to Cosmological simulations requires including periodic boundary conditions. This has been done using Ewald's method (Rybicki, 1986; Hernquist, Bouchet and Suto, 1991). Ewald's method is used to tabulate the correction to the force due to periodic boundary conditions. This correction term is stored on a grid (in relative separation of a pair of particles) and the interpolated value is added to the pairwise force.

Some attempts have been made to combine the high resolution of a tree code with the natural inclusion of periodic boundary conditions in a PM code (Xu, 1995). In this paper we present a hybrid N-Body method that attempts to combine these features. Our approach differs significantly from that of Xu (1995) and we will comment more on the similarities and differences between our approach and theirs in a later section. The basic motivation for attempting such a hybrid code is to improve the dynamic range over which the results are trustworthy. It is similar in spirit to P^3M codes but avoids the variation of speed with the level of clustering.

The plan of the paper is as follows: §2 introduces the basic formalism of both the tree and particle mesh codes. §2.3 gives the mathematical model for splitting the force

⋆ E-mail : jbagla@cfa.harvard.edu

between the two components, and hence is the mathematical model for the treePM code. We analyse errors in force for the treePM code in §3, and use this analysis to fix some parameters, like the transition scale between the tree and the PM part. We fix these parameters to get a fast and accurate configuration for the treePM code. We present some sample results of a treePM simulation run with these parameters. Computational requirements of our implementation of the treePM code are discussed in §4, along with a table of the time this code takes on a workstation for one time step. A discussion of the relative merits of the treePM code vs the TPM code (Xu, 1995) and P^3M is given in §5.

2 TREE + PM = TREEPM

2.1 Tree Code

We use the Barnes and Hut (1986) (BH86 hereafter) tree code. In this code the simulation volume is taken to be a cube. If this is taken to represent the stem of a tree then it is subdivided at each stage into smaller cubes (branches) till we reach the particles (leaves). To construct the tree, we add particles to the simulation volume and subdivide any cell that ends up with two particles. More details can be found in the original paper (Barnes and Hut, 1986).

The force on a particle is computed by adding contribution of other particles or of cells. If a cell is too close to the particle, or if it is too big, we consider the sub cells of the cell in question instead. The decision is made by computing the quantity θ and comparing it with a threshold θ_c:

$$\theta = \frac{d}{r} \leq \theta_c \tag{1}$$

where d is the size of the cell and r is the distance from the particle to the centre of mass of the cell. The error in force increases with θ_c and for large values ($\theta_c \approx 1$), it is important to make sure that no self force is included by mistake (Hernquist, 1987). There are some potentially serious problems associated with using large θ_c, a discussion of these and some remedies is given in Salmon and Warren (1994). In general, it is safe to use $\theta_c \leq 1/\sqrt{3}$. It is also possible to use a completely different approach and use some other criterion, e.g., one can require that the contribution of a given cell to the total force on a particle should be less than some fraction of the total force (Springel and White, 1999). This method does not waste time subdividing cells that do not have many particles, but requires calculation of the force with a very small value of θ_c for the first step.

Irrespective of the criterion used, the number of terms that contribute to the force of a particle is much smaller than the total number of particles for most choices of θ_c or its equivalent, and this is where a tree code gains on a direct particle-particle code.

The performance of a tree code can be improved even further by making use of the fact that the interaction lists† of neighbouring particles are very similar. Thus we can calculate force for a group of particles that are close to each other in one tree walk, and add the interaction terms for within the group by a direct particle sum (Barnes, 1990). As long as the direct sum within the group does not dominate the number of terms, this method leads to a significant improvement in speed of the code. However, as mentioned in Barnes (1990), the acceptance criterion (eqn.1) for the cells has to be modified so that r is measured from the edge of the group rather than the centre of mass of the group. To implement this, we subtract the distance of the particle that is furthest from the centre of mass of the group, from the distance between the centre of mass of the group and the node in question. Thus the new acceptance criterion is:

$$\frac{d}{|r - r_{max}|} \leq \theta_c, \tag{2}$$

where r_{max} is the distance between the centre of mass of the group to the most distant member of the group. The distance between the centre of mass of the cell and the node is r. This is a more conservative criterion than some others, but is easier to implement without sacrificing the accuracy or the efficiency of the code.

We will use the BH86 tree code with the grouping scheme described above for the tree part of the code. We include periodic boundary conditions for computing the short range force on particles near the boundaries of the simulation cube. Another change to the standard tree walk is that we do not follow nodes representing cells that do not have any spatial overlap with the region within the threshold radius for computing the short range force.

2.2 Particle Mesh Code

A particle mesh code (PM hereafter) is the obvious choice for computing long range interactions. Much has been written about the use of PM codes in cosmological simulations (e.g., see Hockney and Eastwood, 1988) so we will not go into the details here. Basically, the PM code adds the construct of a regular grid to the distribution of particles. The density field represented by particles is interpolated onto the grid points and the Poisson equation is solved in Fourier space. The force is then interpolated back to the positions of particles. Use of a grid implies that forces are not accurate at the scale of the grid, or at smaller scales. A discussion of errors in force in a PM code can be found in Efstathiou et al. (1985) and elsewhere (Bouchet and Kandrup, 1985; Bagla and Padmanabhan, 1997). The error in force can be very large at small scales but it drops to an acceptable number beyond a few grid cells, and is vanishingly small at large scales. As we will use the PM code only for long range force, this should not be a problem.

We use the Cloud-in-Cell weight function for interpolation. We solve the Poisson equation using the natural kernel, $-1/k^2$; this is called the poor man's Poisson solver (Hockney and Eastwood, 1988). We compute the gradient of the potential in Fourier space. This can be done in real space to reduce memory requirement.

We now turn to the question of combining the tree and the PM code.

2.3 TreePM Code

We wish to split the inverse square force into a long range force and a short range force. We compute the long range

† Interaction list is the complete list of nodes, cells and particles, that contribute to the force acting on a particle.

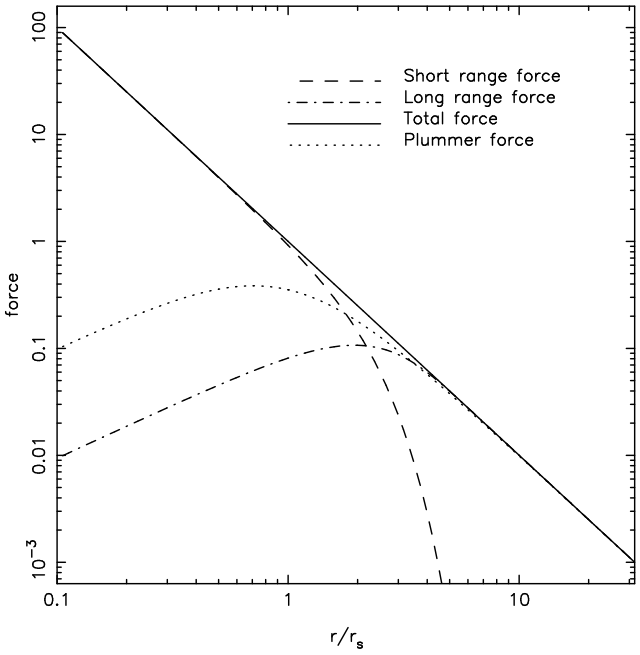

Figure 1. This figure shows the long and the short range force as a function of scale. The long range force (dot-dashed line) peaks around $2r_s$ and drops linearly towards smaller scales. The short range force (dashed line) drops sharply at scales beyond $3r_s$ and falls two orders of magnitudes below the true force at $5r_s$. The thick line is the sum of the short and the long range force. For comparison, we have included the Plummer force (dotted line) with r_s as the softening length.

force in the Fourier space and the short range force in real space. Following Ewald's method (Ewald, 1921), the gravitational potential can be split into two parts in Fourier space:

$$\begin{aligned}
\varphi_k &= -\frac{4\pi G\varrho_k}{k^2} \\
&= -\frac{4\pi G\varrho_k}{k^2}\exp\left(-k^2r_s^2\right) - \frac{4\pi G\varrho_k}{k^2}\left(1-\exp\left(-k^2r_s^2\right)\right) \\
&= \varphi_k^l + \varphi_k^s \\
\varphi_k^l &= -\frac{4\pi G\varrho_k}{k^2}\exp\left(-k^2r_s^2\right) \qquad (3) \\
\varphi_k^s &= -\frac{4\pi G\varrho_k}{k^2}\left(1-\exp\left(-k^2r_s^2\right)\right) \qquad (4)
\end{aligned}$$

where φ^l and φ^s are the long range and the short range potentials, respectively. The splitting is done at the scale r_s. G is the gravitational coupling constant and ϱ is density. The short range potential and force are evaluated in real space, using the following expressions.

$$\begin{aligned}
\varphi^s(\mathbf{r}) &= -\frac{G}{r}\mathrm{erfc}\left(\frac{r}{2r_s}\right) \\
\mathbf{f}^s(\mathbf{r}) &= -\frac{G\mathbf{r}}{r^3}\left(\mathrm{erfc}\left(\frac{r}{2r_s}\right) + \frac{r}{r_s\sqrt{\pi}}\exp\left(\frac{r^2}{4r_s^2}\right)\right) \qquad (5)
\end{aligned}$$

Here, erfc is the complementary error function. The expression for the long range force in real space is:

$$\mathbf{f}^l(\mathbf{r}) = -\frac{G\mathbf{r}}{r^3}\left(\mathrm{erf}\left(\frac{r}{2r_s}\right) - \frac{r}{r_s\sqrt{\pi}}\exp\left(\frac{r^2}{4r_s^2}\right)\right). \qquad (6)$$

We have plotted the long range (eqn.(6)) and the short range force (eqn.(5)) as a function of r/r_s in fig.1 to show their dependence on scale. The total force is shown by the thick line. The short range force (dashed line) closely follows the total force up to about $2r_s$ and then falls rapidly, its magnitude falls below 1% of the total force by $5r_s$. The long range force reaches a peak around $2r_s$. It makes up most of the total force beyond $3.5r_s$. It falls linearly with scale below $2r_s$, becoming negligible below $r_s/2$.

For comparison, we have also plotted the Plummer force ($\mathbf{f} = -\mathbf{r}/(r^2+r_s^2)^{3/2}$) with r_s as the softening length (dotted line). The difference between the true force and the Plummer force increases very slowly towards small scales. This implies that any error in the long range force will contribute significantly to the error at small scales. The slow rate of convergence at large scales also implies that the short range correction falls below 1% of the total force only around $6.5r_s$ compared to $5r_s$ for the split achieved using Ewald's method.

Eqns.(3,5) provide a prescription for computing the long range and the short range force independently. As mentioned above, we compute the short range force in real space, using the tree code, and the long range force is computed in Fourier space using a PM code. Both of these are computed for every particle in the simulation.

Evaluation of spacial functions for calculating the short range force can be time consuming. To save time, we compute an array containing the scalar part of the short range force. The force between any two objects, particle-cell or particle-particle, is given by the nearest array element multiplied it by the vector $\mathbf{r}$.

3 ERROR ESTIMATION

In this section we will study errors in force introduced by various components of the treePM code.

We start by estimating the error in force due to one particle. We focus on small scales as the error in force at large scales is negligible, well below 1%. In this range of scales, we can take the true force to vary as inverse square of the distance. To estimate the errors, we place a particle randomly in a cell and compute the force at 10^4 random positions in the region around the particle. The size of the cubical volume is 128^3 and we use periodic boundary conditions. We repeat this process 50 times. This is done for two values of the transition scale, $r_s = 1$ and $r_s = 4$.‡ The results are plotted in fig.2. Magnitude of force is plotted as a function of distance from the particle (We used the Plummer softening with softening length $\epsilon = 0.1$). The true force, which coincides with that computed using $r_s = 4$, is plotted as the thick line. The dashed line shows the average force for $r_s = 1$. The dispersion around this is very small, it is less than 1.5% over the entire range of scales and is thus comparable to the thickness of the line. The main difference between the true force and that computed using $r_s = 1$ is a systematic shift towards lower force around the transition scale. The reason for this shift is the incomplete coverage of the long range force in $k-$space. The dispersion for the larger transition scale is well below 1%.

‡ We use the size of one grid cell as the unit of length.

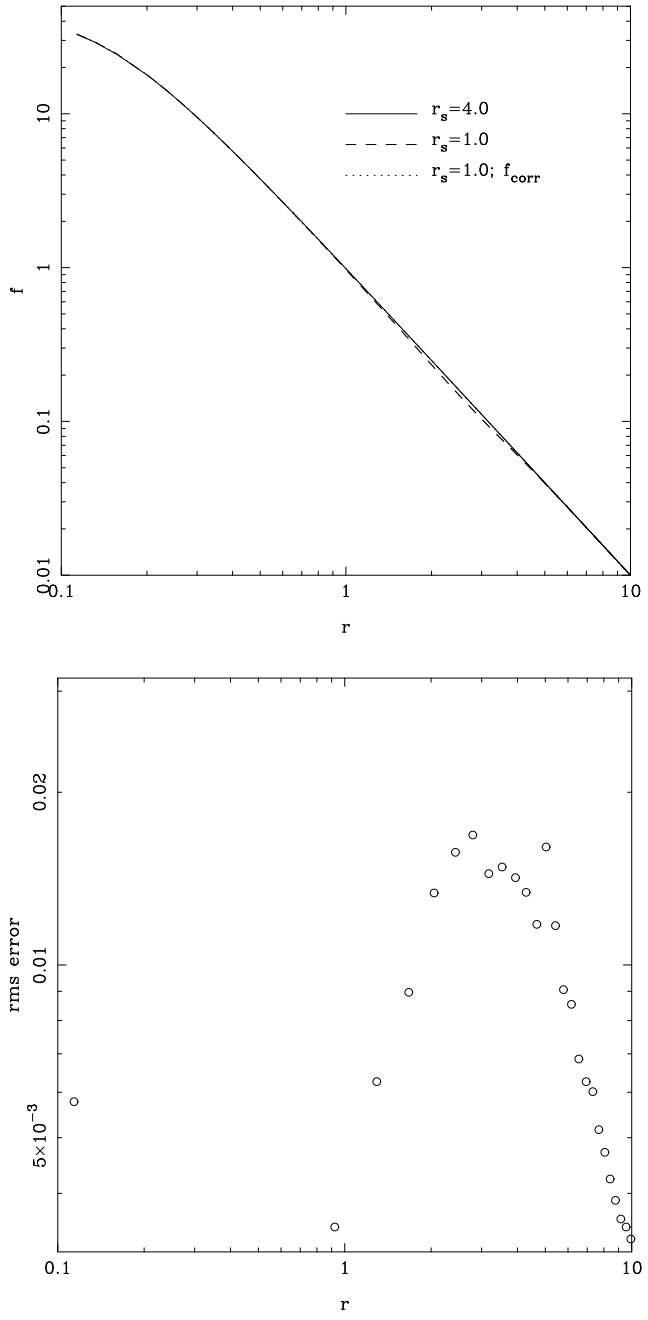

Figure 2. The top panel shows the magnitude of force as a function of scale. The thick line marks the force computed for $r_s = 4$, this coincides with the true force. The dashed line shows the force computed using $r_s = 1$. It matches well with the true force at all the scales except for a narrow range of scales above r_s. As the dispersion around the average shown here is negligible, we add an extra correction and plot the average as the dotted line – this coincides with the true force. The lower panel shows the dispersion around the average plotted in the top panel. Circles show the rms value of the fractional error as a function of scale. As can be seen here, this remains below 2% for the entire range of scales.

This figure suggests that it is preferable to use a large transition scale for better accuracy. However, using a large r_s increases the search radius for adding the short range correction and takes away the advantage in terms of speed. Given that the dispersion is smaller than the systematic shift, it is reasonable to add an extra "correction term" to the short range force to take care of this shift. We estimate the correction by taking the difference between the true force and the average force shown as the dashed line in fig.2. This difference is below 7% and is larger than the dispersion around the mean only in the range $1 \leq r/r_s \leq 5$. As this is a subset of the range for which we compute the short range force, we add this correction term to the array containing the short range force. Force with this correction put in is plotted as a dotted line in the same panel. It coincides with the thick line through the entire range of scales.

Fig.2b shows the fractional error as a function of scale. The fractional error is defined as:

$$\frac{\delta f}{f_0} = \frac{|\mathbf{f} - \mathbf{f}_0|}{|\mathbf{f}_0|} \tag{7}$$

We show the fractional error for $r_s = 1$ with the correction for the shift in force put in. The dispersion is small and remains below 2% for the entire range of scales.

Fig.2 shows that the errors in force are very small for a large range in scales. Errors, as expected, are localised around the transition scale. Almost all the errors arise due to anisotropies in the PM force. The errors in the PM force increase as we approach small scales, but the contribution of the PM force to the total force falls sharply below $2r_s$ and hence the errors also drop rapidly. This is what gives rise to the peak in error around this scale. Nevertheless, the rms error is below 2% in both the cases and the average error is very close to zero throughout. The treePM code gives optimum performance for small r_s as the search radius for computing the short range force is proportional to this scale. As the errors are acceptable for $r_s = 1$, we will use this transition scale in the final configuration. Reducing r_s further leads to rapid increase in errors.

In plotting fig.2, we added the small scale force to the long range force at all scales. However, this is not necessary as beyond some scale, the contribution of small scale force to the total force drops to a negligible fraction of the total force. We will call the scale upto which we add the small scale force as r_{cut}. The short range force, with the correction put in, is just below 1% of the total force at $r_{cut} = 5r_s$. We choose this value of r_{cut} for the treePM code as at this distance, even a 100% error will not contribute more than 1% to the total error budget. The execution time, for a fixed θ_c, increases with r_{cut} so this should be as small as the accuracy requirements allow.

The last factor that we need to study is the variation of error with θ_c, the threshold opening angle. The answer in this case is not as straightforward as it is for a normal BH tree code. For a given r_{cut}, the variation of error is still monotonic and it increases with increasing θ. However, this variation is not smooth and comes in discrete jumps. The reason being that the dominant contribution to the error comes from the cells that are close to r_{cut}, as the contribution of the small scale force varies very rapidly around this scale. Therefore, it is better to avoid using a large θ_c and a large r_s. Thus we have to strike a balance between

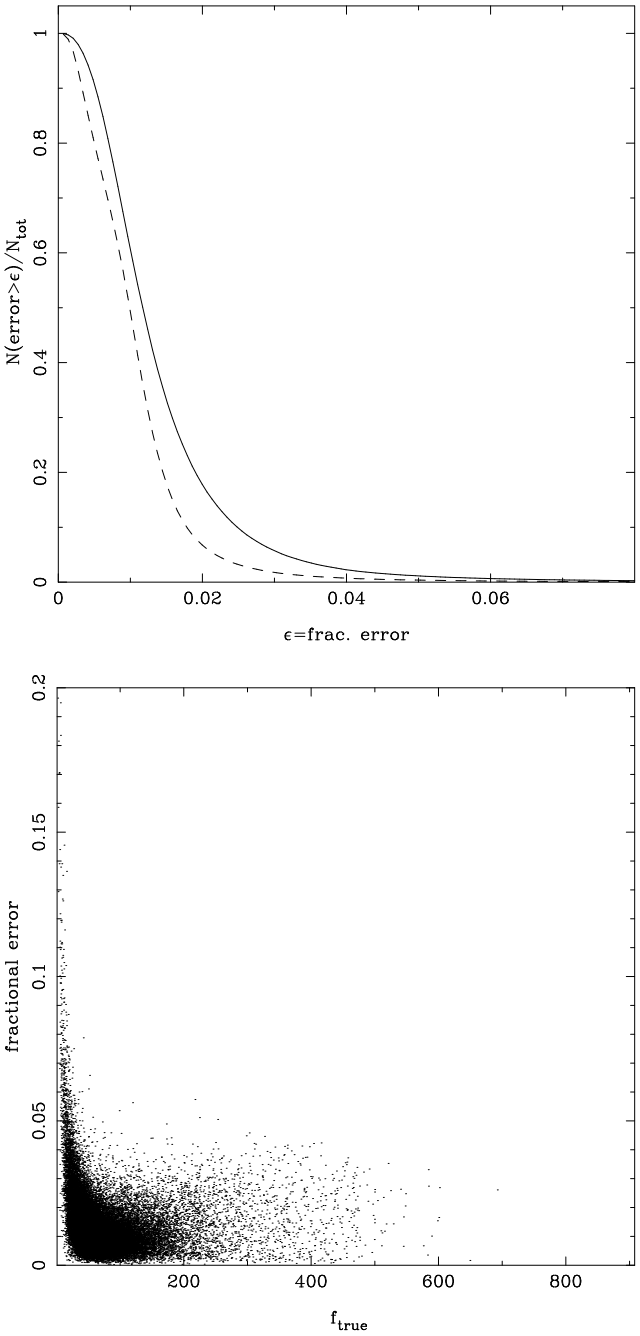

Figure 3. This figure shows the distribution of errors. The top panel shows the variation of the fraction of particles with error greater than a threshold, as a function of the threshold. Thick line marks the error for a homogeneous distribution of particles and the dashed line shows the same for a clumpy distribution. These errors were measured with respect to a reference force, determined with a very conservative value of r_s, r_{cut} and θ_c. This panel shows that 99% of the particles have fractional error in force that is less than 5% for the homogeneous distribution and around 3% for the clumpy distribution. The lower panel shows the fractional error of a subset of particles as a function of the net force acting on that particle. These particles were picked from the homogeneous distribution used in the upper panel. It is clear that the error is small for all except a few particles with a very small net force acting on them. These particles form a small fraction of the total, less than 1% have error greater than 5%.

the anisotropies from the long range force for small r_s and large errors for the short range force from particles and cells around r_{cut} for large r_s.

Another factor that we have to weigh in is that the execution time is small for large θ_c and small r_{cut}. Given these considerations, the obvious solution is to choose the smallest r_s and the largest θ_c that gives us a sufficiently accurate force field.

It is important to estimate the errors in a realistic situation, even though one expects that errors will not add up coherently in most situations. This is also important because none of the tests outlined above have checked for errors induced by the tree approximation. We test errors for two distributions of particles: a homogeneous distribution and a clumpy distribution. For the homogeneous distribution, we use randomly distributed particles in a box. We use 262144 particles in a 64^3 box for this distribution. We compute the force using a reference setup ($r_s = 4$, $r_{cut} = 6r_s$, $\theta_c = 0$) and the setup we wish to test ($r_s = 1$ with force correction, $r_{cut} = 5r_s$, $\theta_c = 0.5$). We compute the fractional error for each particle, we prefer this to the error in magnitude of the force as this checks for error in the direction of force as well. Fig.3a shows the distribution of fractional errors. We have drawn the number of particles with error greater than ϵ as a function of ϵ (thick line). Force error for 99% of particles is less than 5%. Almost all the particles with a large force error are those with a small net force. This is shown in fig.3b which shows the fractional error in force on a random subset of particles as a function of the "true" or reference force. Almost all the particles have a very low error except for particles that have a very small net force.

Results for the clumpy distribution of particles are shown by the dashed line in fig.3a. We used the output of a CDM simulation (fig.4a) run with the treePM code. The typical errors in this case are much smaller, as in the case of tree code (Hernquist, Bouchet and Suto, 1991). Force error for 99% of particles is around 3%, as compared to 5% for the homogeneous distribution.

We end this section with a brief comparison of the treePM code with a PM code. We ran a simulation of the sCDM model (262144 particles, $64h^{-1}$Mpc box) with a PM code (Bagla and Padmanabhan, 1997) and with the treePM code discussed here. Fig.4 shows a slice from these simulations; fig.4a shows the simulation with the treePM code and fig.4b shows the same for a PM code. The large scale structures are the same in the two but there are significant differences at small scales. The halos are much more compact in the treePM simulation, and large halos show more substructure. These differences are also clear in the two point correlation function $\bar{\xi}(r)$ plotted in fig.5. The thick line shows the correlation from the treePM simulation and the dashed line shows the same for the PM simulation. As expected from fig.4 and from general considerations, the correlation function in the treePM simulation matches with that from the PM simulation at large scales, but at small scales, the treePM simulation has a higher correlation function.

To show that this code follows the linear evolution of density perturbations correctly, we have plotted the power spectrum of density fluctuations at many epochs in fig.6. We have scaled the amplitude of power spectrum by the linear growth rate and plotted the quantity $\Delta(k)/a$ =

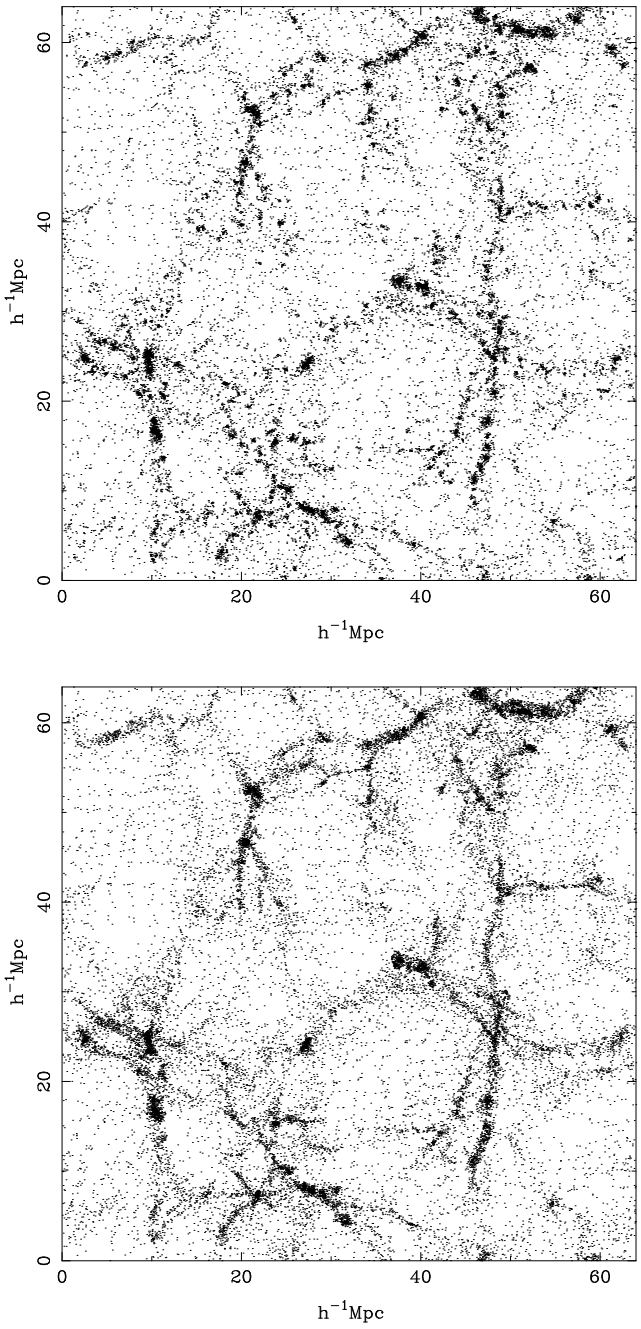

Figure 4. This figure shows a slice from a simulation of the sCDM model. The top panel shows the slice from the treePM simulation. For comparison, we have included the same slice from a PM simulation of the same initial conditions. The large scale structures are the same in the two but there are significant differences at small scales. The halos are much more compact in the treePM simulation, and large halos show more substructure. This is to be expected because of the superior resolution of the treePM code.

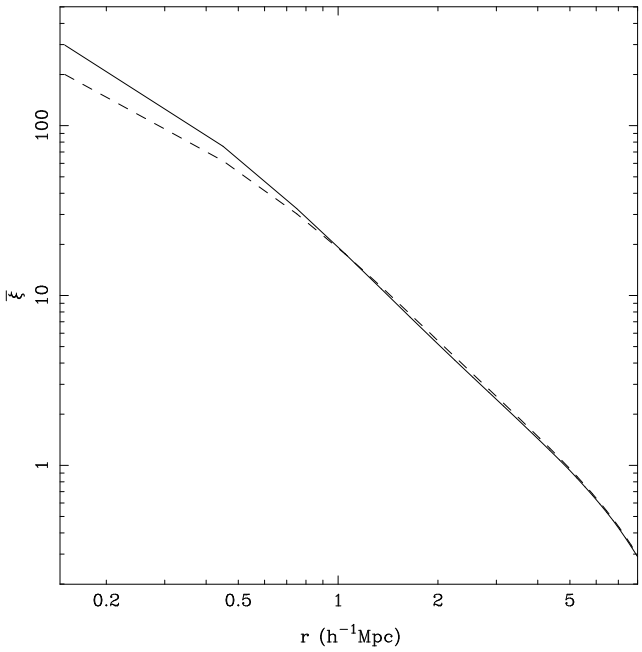

Figure 5. This figure shows the averaged correlation function $\bar{\xi}(r)$ as a function of scale. The thick line shows this quantity for the treePM simulations and the dashed line shows the same for the PM simulation. These two match at large scales but the PM simulation underestimates the clustering at small scales.

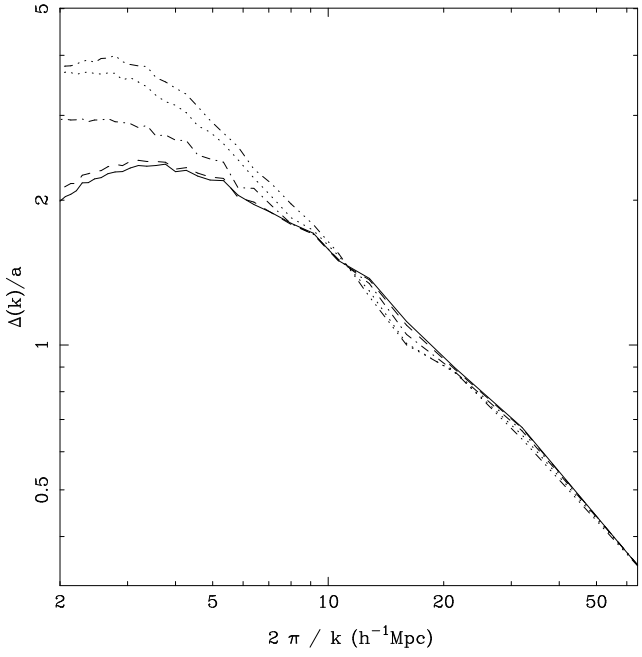

Figure 6. This figure shows the variation of power spectrum in the treePM simulation shows in fig.4. We have plotted the quantity $\Delta(k)/a$, where $\Delta^2(k) = k^3P(k)/2\pi^2$ as a function of $2\pi/k$. The thick line shows the initial power spectrum at $a = 0.1$. The other lines are: dashed line for $a = 0.2$, dot-dashed line for $a = 0.4$, dotted line for $a = 0.7$ and dot-dot-dashed line for $a = 1.0$. Linear evolution is followed correctly, shown by overlapping of all the curves at large scales. At small scales, non-linear clustering leads to a faster than linear growth in the power spectrum.

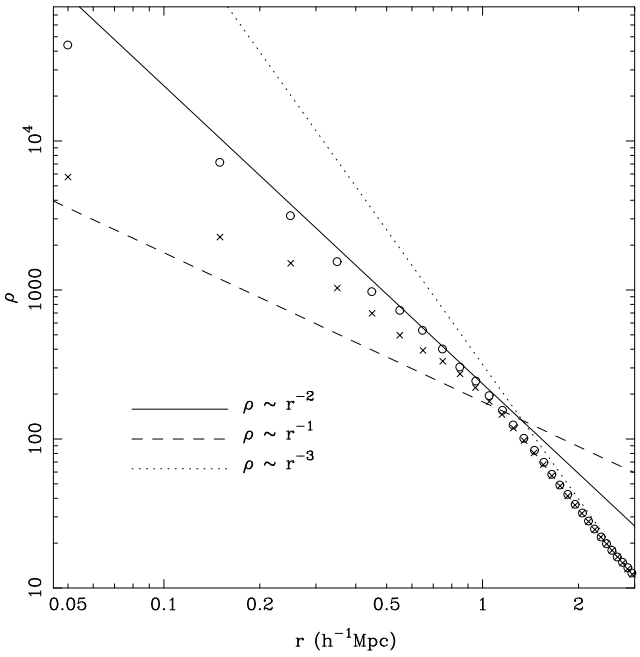

Figure 7. This figure shows the density profile of a rich cluster taken from the simulation of sCDM mentioned above. The average density within a given radius is plotted as a function of radius. Density in the treePM simulation ○ lies well above that in the PM simulation ×. Both curves match at scales larger than one grid length. Slopes of $1/r$, $1/r^2$ and $1/r^3$ are marked for reference.

$\sqrt{k^3P(k)/2\pi^2a^2}$ as a function of $2\pi/k$. The thick line shows the initial power spectrum at $a = 0.1$. The other lines are: dashed line for $a = 0.2$, dot-dashed line for $a = 0.4$, dotted line for $a = 0.7$ and dot-dot-dashed line for $a = 1.0$. It is clear that the linear evolution is followed correctly, shown by overlapping of all the curves at large scales. At small scales, non-linear clustering leads to a faster than linear growth in the power spectrum.

Lastly, we show density profile of a rich cluster from the simulations shown in fig.4. We have plotted average density ($\rho(r) = 3M(r)/4\pi r^3$) as a function of r for the rich cluster from the treePM and the PM simulation in fig.7. The density in the treePM simulation is marked by open circles, whereas crosses mark the density in the PM simulation. The density near the centre is much higher in the treePM simulation, as expected. At larger scales, where the differences of resolution do not make much difference, density in both the simulations converges to the same value. Higher resolution in the treePM simulation also leads to the steeper inner slope for the peak. This cluster has about 10^3 particles inside the virial radius.

In many clusters, as is evident from fig.4, the higher resolution allows us to resolve more substructure in the treePM simulation than in the PM simulation.

4 COMPUTATIONAL RESOURCES

In this section, we describe the computational resources required for the present implementation of the treePM code. Given that we have combined the tree and the PM code, the memory requirement is obviously greater than that for either one code. We need four arrays for the PM part, the potential and the force. The rest is exactly the same as a standard Barnes and Hut tree code. With efficient memory management, we need less than 160MB of RAM for a simulation with 128^3 particles in a 128^3 mesh for most part. In absence of memory management, this requirement can go up to 250MB.

Table 1. Time taken by the code, per time step per particle. Column 1 lists the number of particles. Column 2 and 3 list the time taken (per time step per particle) by the treePM code for an unclustered and a clustered particle distribution. Column 4 lists the same number for a tree code for an unclustered distribution of particles. All the times are in milli seconds.

$N_{particle}$	time (ms) treePM unclustered	time (ms) treePM clustered	time (ms) tree unclustered
32768	0.57	0.58	2.94
262144	0.78	0.79	3.75
2097152	1.22	1.24	6.03

Table 1 lists the time required per time step per particle for three values of the number of particles. These were run on a 533MHz Alpha workstation (EV5) and compiled with the native F90 compiler. Column 1 lists the number of particles and col.2 lists the time per step per particle for an unclustered distribution. This number increases slightly faster than the number of particles.

Column 3 of table gives the same number for a highly clustered particle distribution, similar in clustering strength to that shown in fig.4. Column 4 lists the time per step per particle taken by the tree code for the particle distribution used in col.2. It is clear that the treePM code is faster than the tree code by a factor of about 4.5.

The performance of this code can be improved further by including features like individual time steps for particles. We will not go into that optimisation here because we want to focus on the speedup achieved by combining the tree and the PM codes as outlined in earlier sections.

5 COMPARISON WITH OTHER METHODS

The treePM code retains the accuracy of the tree code while speeding up the force calculation by a factor of 4.5 or more. The codes that provide comparable performance are the P^{3}M (Efstathiou et al, 1985; Couchman, 1991) codes and the TPM code (Xu, 1995). Following subsections compare treePM with the other two codes.

5.1 P^{3}M and AP^{3}M

There are two main differences between P^{3}M (Efstathiou et al, 1985; Couchman, 1991) codes and the treePM code presented here. One is that most P^{3}M codes use the natural cutoff provided by the grid for the long range force. This implies that the long range force has the form of Plummer force with a softening scale comparable to grid size. In contrast, we use Ewald's method for effecting the separation between

the long and the short range force. Fig.1 shows that the Plummer force declines much more slowly at small scales, thus anisotropies due to the mesh will lead to anisotropies in force at scales comparable to grid scale.

The second difference is that the small scale force is added for each pair of particles with separation smaller than some r_{cut}. This process is of order $O(Nnr_{cut}^3(1+\bar{\xi}(r_{cut})))$, where N is the number of particles in the simulation, n is the number density of particles and $\bar{\xi}(r_{cut}) = 3J_3(r_{cut})/r_{cut}^3$. At early times this reduces to $O(Nnr_{cut}^3)$, but at late times, when the density field has become highly non-linear ($\bar{\xi}(r_{cut}) \gg 1$), it becomes $O(Nnr_{cut}^3\bar{\xi}(r_{cut}))$. Thus, as the density field becomes more and more non-linear, the number of operations required for computing the short range force increase rapidly. The number of operations required for adding the short range correction in the tree code varies much more slowly: $O(N\log(nr_{cut}^3(1+\bar{\xi}(r_{cut}))))$. The linear and the non-linear limits of this expression are $O(N\log(nr_{cut}^3))$ and $O(N\log(nr_{cut}^3\bar{\xi}(r_{cut})))$, respectively. Thus the variation in the number of operations with increase in clustering is much less for treePM code a P^3M code. The problem is not as severe as outlined for the Adaptive P^3M code (Couchman, 1991) but it still persists. Therefore the treePM code has a significant advantage over the P^3M and AP^3M code for simulations of models where $\bar{\xi}(r_{cut})$ is very large.

In turn, P^3M codes have one significant advantage over treePM, these require much less memory. This gives P^3M codes an advantage on small machines and for simulations of models where $\bar{\xi}(r_{cut}) \leq 1$.

5.2 TPM

The difference between the treePM code presented here and the TPM code (Xu, 1995) are as follows:

- The TPM code uses the natural smoothing by the grid to truncate the long range (PM) force.
- The treePM treats all the particles on an equal footing, we compute the short range and the long range force for each particle. In the TPM code, the short range force is computed only for particles in the high density regions.[§] The tree particles are further subdivided into groups. The interaction of particles within a group is computed at high spatial resolution and the force due to particles outside the group is given by the PM part of the code. This has the advantage of parallelisation by assigning different groups to different nodes in a parallel machine.

Our implementation of the treePM code thus has a cleaner mathematical model with only one parameter, the transition scale between the tree and the PM components, r_s. In our view, this makes error estimation fairly simple as the number of configurations in which the PM and the tree code interface is very limited.

[§] This is a simple description of their criteria for deciding whether a given particle is a tree particle or a PM particle (see §2.2 of Xu (1995)). The short range force is computed only for the tree particle.

Parallelising the treePM code is similar to parallelising a tree code, and the algorithms used for parallelising tree codes can be used with the treePM code without any significant modification. Thus the problem of parallelising the treePM code can be mapped to already solved problems.

The tree part is the time consuming part of the treePM code so we will focus on the possibility of parallelising that. The Barnes and Hut tree code has been parallelised using the load balancing scheme suggested by Salmon (1990). This scheme was developed further and a discussion of an implementation using orthogonal recursive bisection (Dubinski, 1996). Such a scheme can be easily generalised to the treePM code, with an added advantage in that the information to be shared amongst different nodes is reduced because of the small search radius in which we add the short range force. The only component of the treePM code that needs to be changed for this is the grouping algorithm, which should not present serious problems.

6 SUMMARY

We have described a hybrid technique for carrying out large N-Body simulations to study formation and evolution of the large scale structure in the Universe. This code, called the treePM code, is a combination of the Barnes and Hut (1986) tree code and the Particle-Mesh code. As shown above, this code combines the speed of PM simulations and the automatic inclusion of periodic boundary conditions with the high resolution of tree codes. Analysis of errors shows that the error budget of this code is better than other codes with comparable resolution. This code provides a significant gain in speed over the tree code while retaining the high resolution. The small search radius for small scale correction should make it easier to parallelise this code, as compared to a tree code.

Acknowledgement

I would like to thank Rupert Croft, Lars Hernquist, Shiv Sethi and Volker Springel for insightful comments and discussions.

REFERENCES

Bagla J.S. and Padmanabhan T. 1997, Pramana – Journal of Physics 49, 161
Barnes, J.E. 1990, J.Comp.Phys. 87, 161
Barnes J. and Hut P. 1986, Nature 324, 446
Bouchet F.R. and Kandrup H.E. 1985, ApJ 299, 1
Bouchet F.R. and Hernquist L. 1988, ApJS 68, 521
Couchman H.M.P. 1991, ApJL 368, L23
Dubinski J. 1996, New Astronomy 1, 133
Efstathiou G., Davis M., Frenk C.S. and White S.D.M. 1985, ApJS 57, 241
Ewald P.P. 1921, Ann.Physik 64, 253
Hernquist L. 1987, ApJS 64, 715
Hernquist L., Bouchet F.R. and Suto Y. 1991, ApJS 75, 231
Hockney R.W. and Eastwood J.W. 1981, *Computer Simulations using Particles*, (New York: McGraw Hill)

Rybicki G.B. 1986, in *The Use of Supercomputers in Stellar Dynamics*, ed. P.Hut and S.McMillan (Berlin: Springer), p.175
Salmon J. 1990, PhD Thesis, *Parallel Hierarchical N-Body Methods*, California Institute of Technology.
Salmon J.K. and Warren M.S. 1994, J.Comp.Phys. 111, 136
Springel V. and White S.D.M. 1999, Preprint
Xu G. 1995, ApJS 98, 355
Zel'dovich Ya.B. 1970, A&A 5, 84